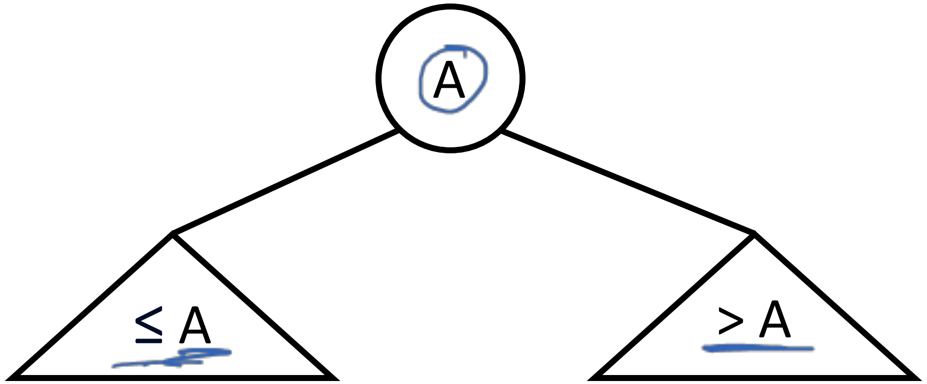
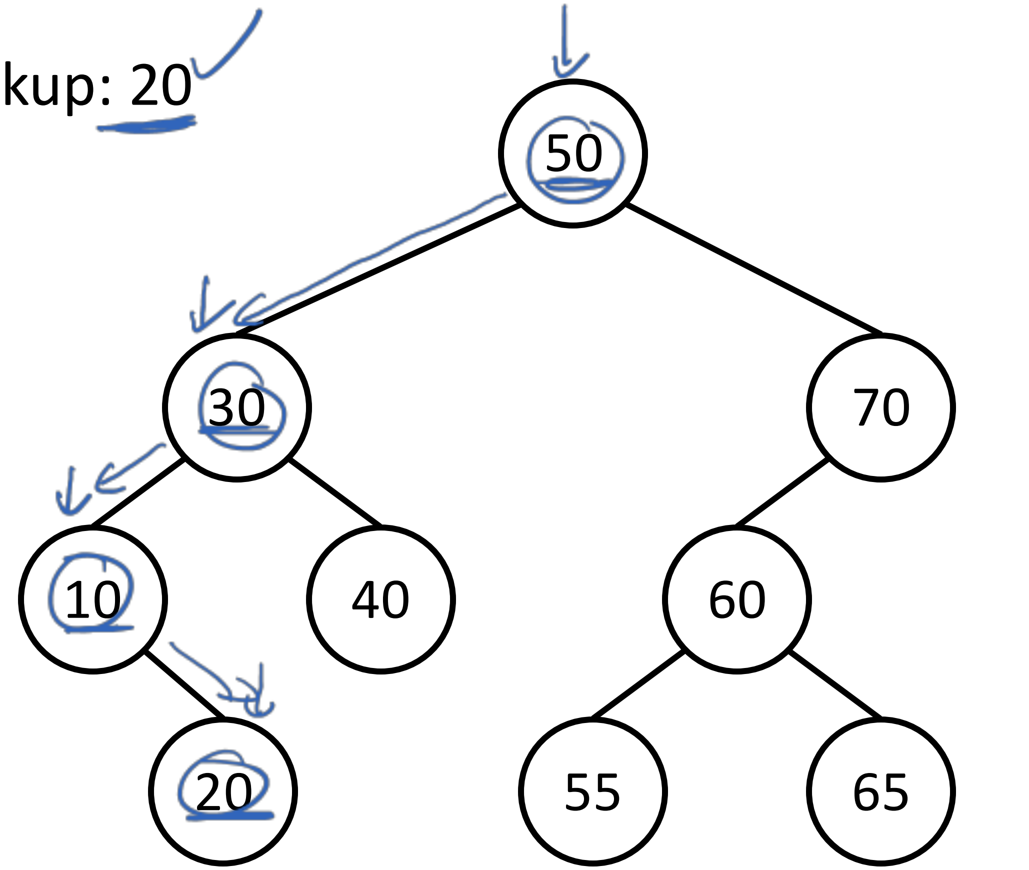
Binary Search Trees Review

Binary Search Trees (BSTs)



Lookup: 20



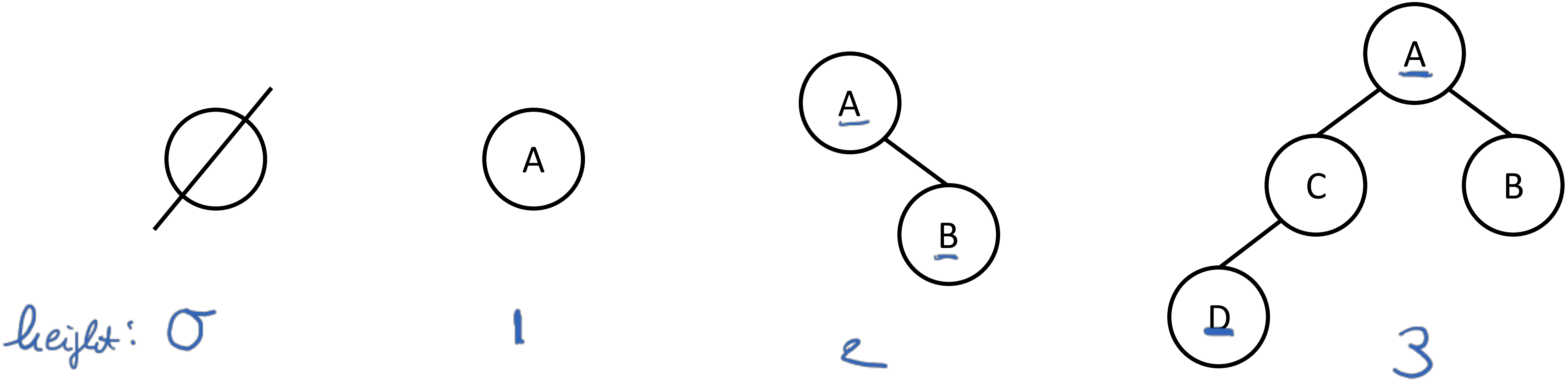
Every node has:

- 0 or 1 parent
- 0, 1, or 2 children

Height of a Tree

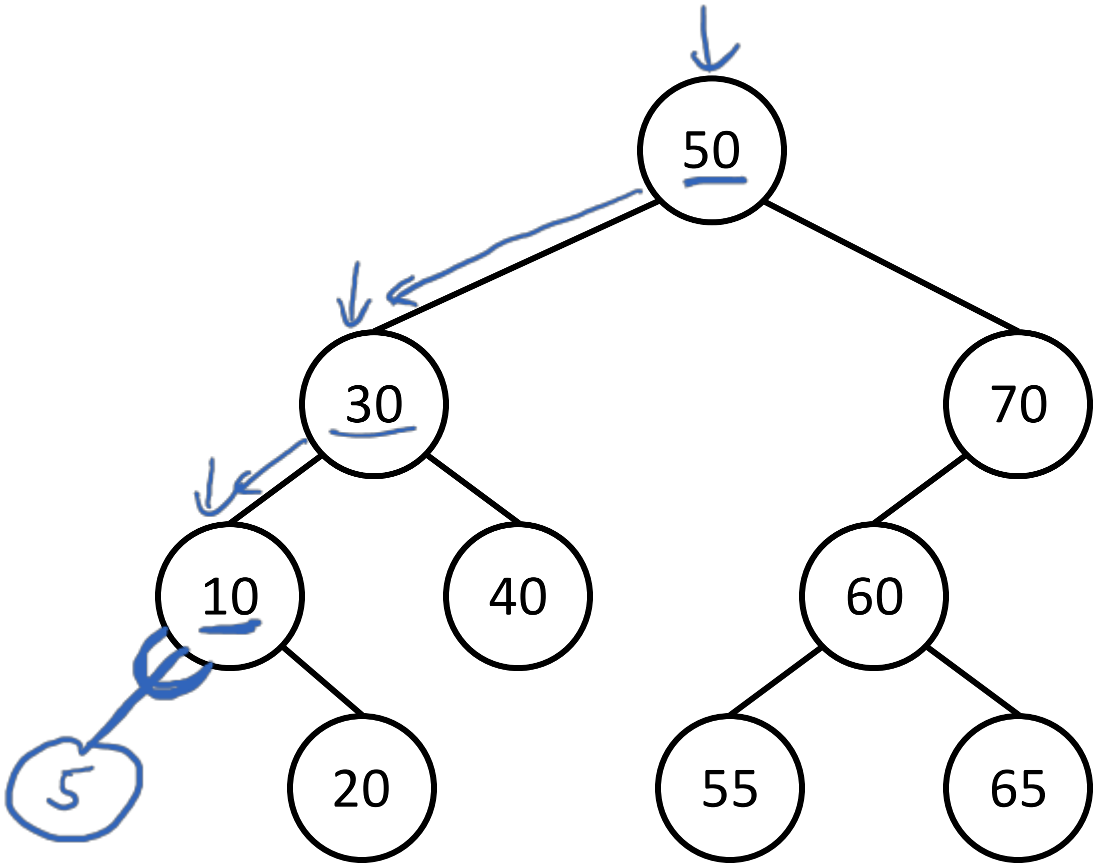
edges

The height of a tree is the number of nodes from the root to the tree's deepest leaf.



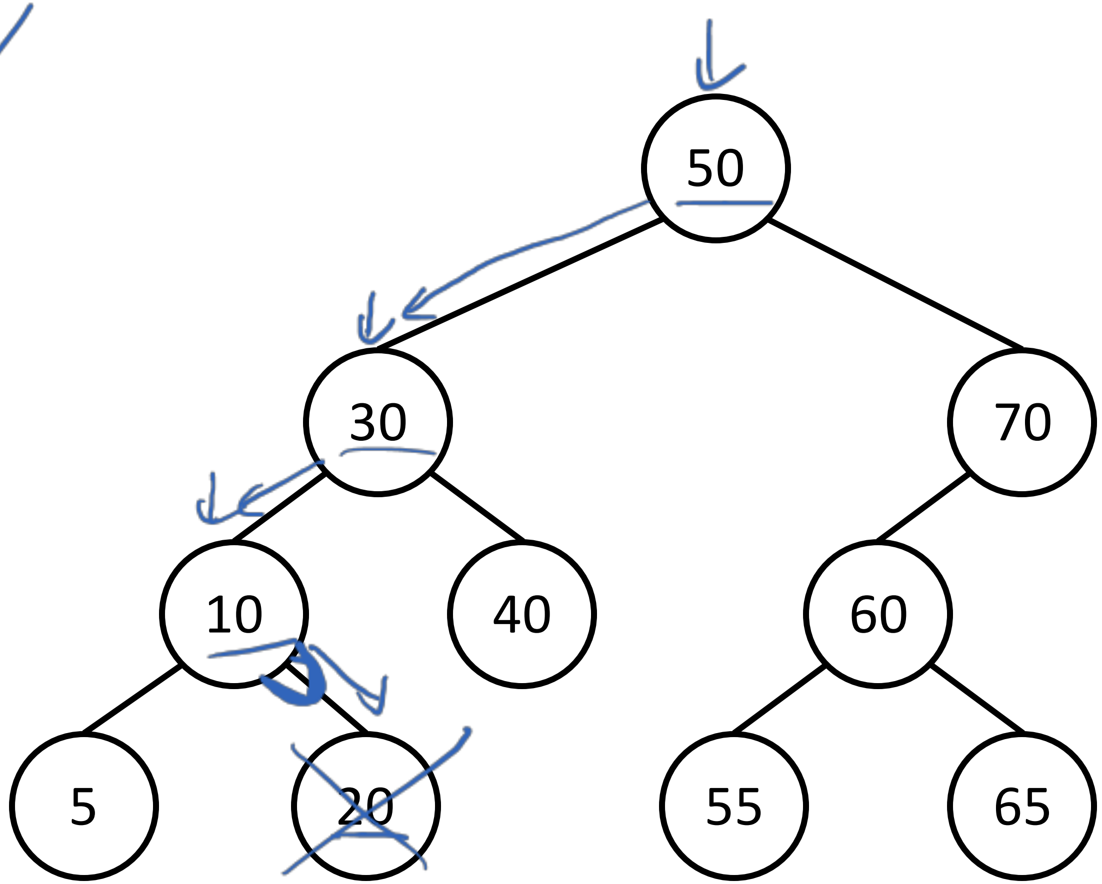
Inserting Value Into Tree

Insert: 5



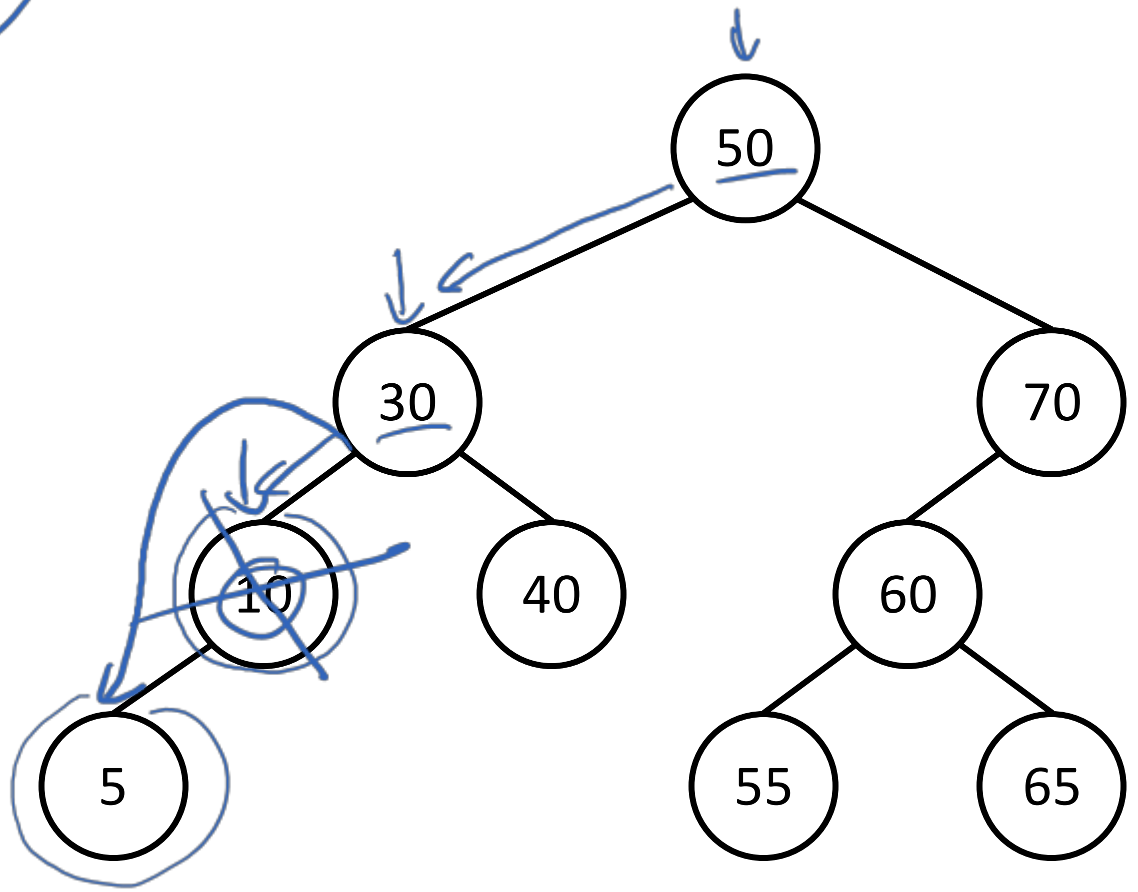
Deleting Value From Tree: Leaf Nodes

Delete: 20



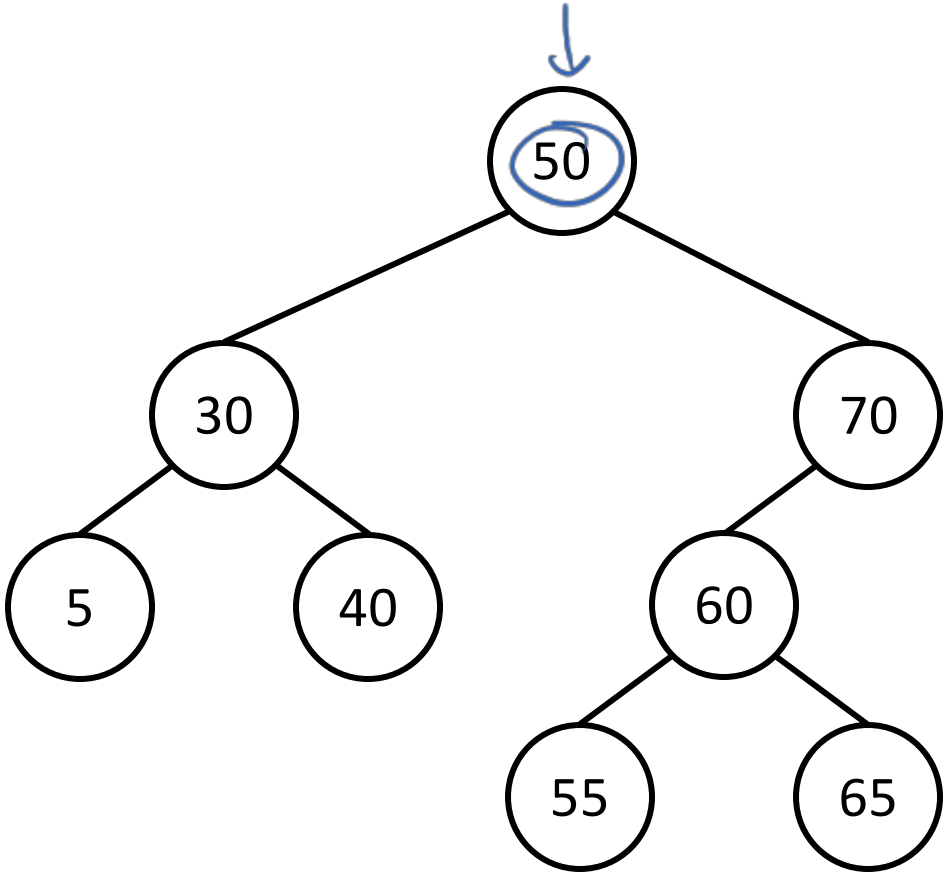
Deleting Value From Tree: One Child

Delete: 10

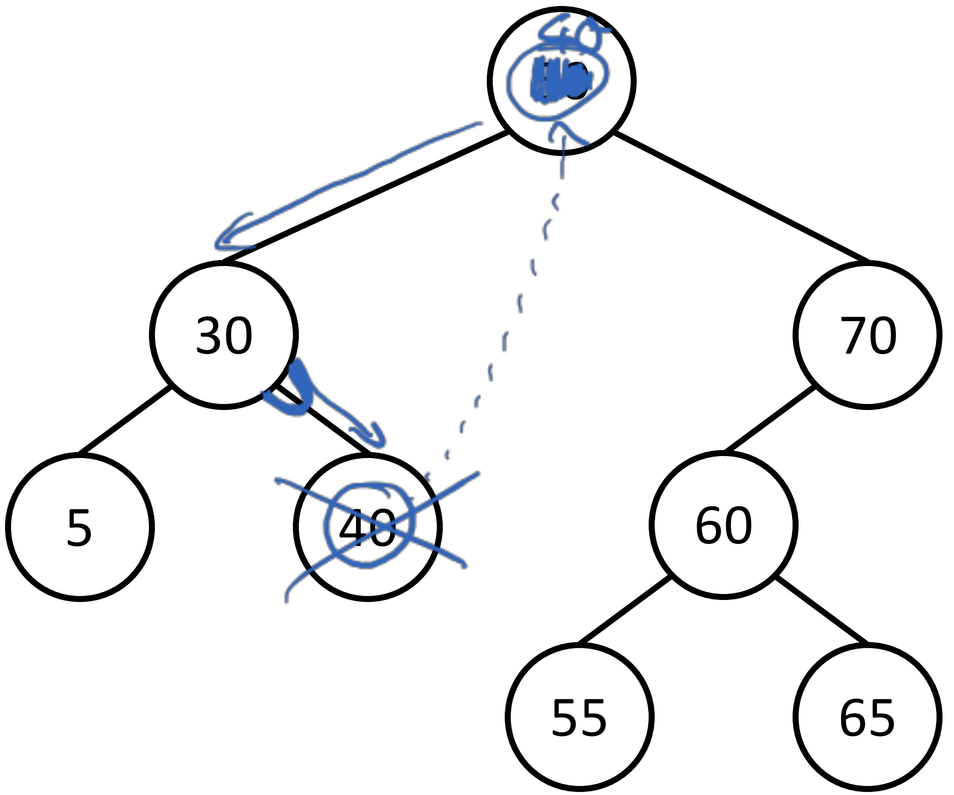


Deleting Value From Tree: Two Children

Delete: 50



Deletion With In-Order Predecessor



Complexities: Search, Insert, and Delete

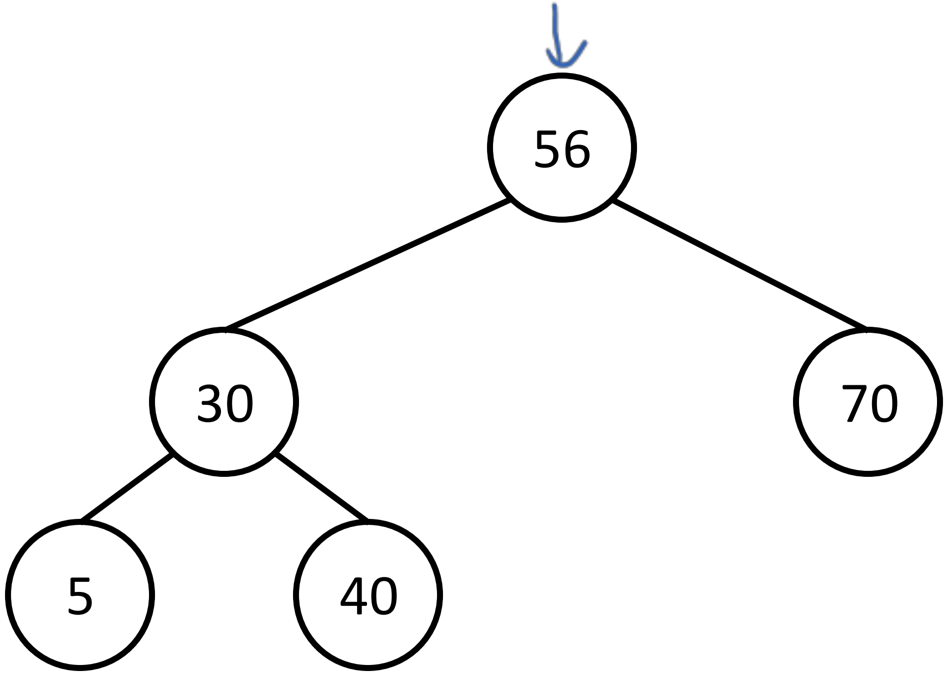
N – number of nodes in the tree

H – height of the tree

Search: $O(H)$

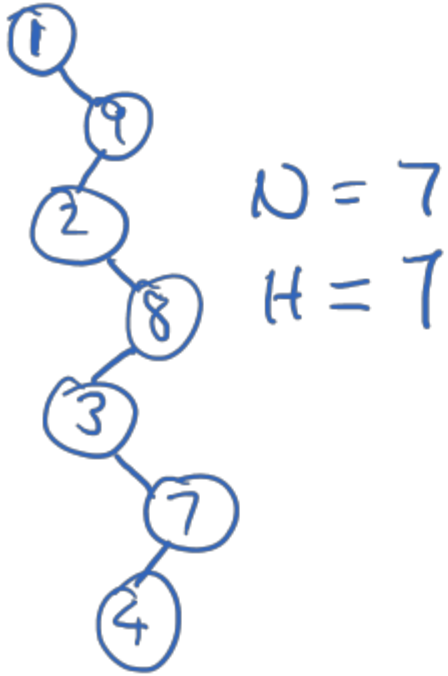
Insertion: $O(H)$

Deletion: $O(\cancel{2}H)$

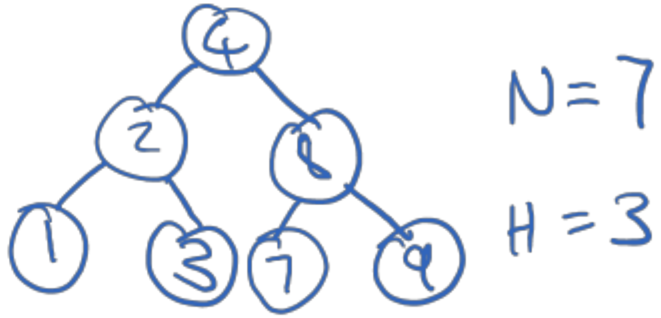


H vs N: Insertion Sequence

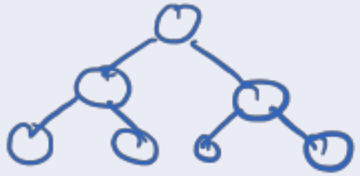
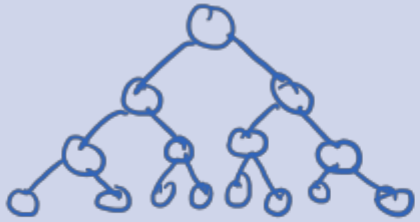
insert: 1, 9, 2, 8, 3, 7, 4



insert: 4, 2, 8, 1, 3, 7, 9



H vs N

H (height)	tree shapes	max N (# of nodes)
0	∅	0
1		1
2		3
3		7
4		15
		⋮

$$N = 2^H - 1$$

Complexities Revisited

N – number of values (nodes) in the tree

H – height of the tree

$O(H)$

$N = 2^H - 1$

$N + 1 = 2^H$

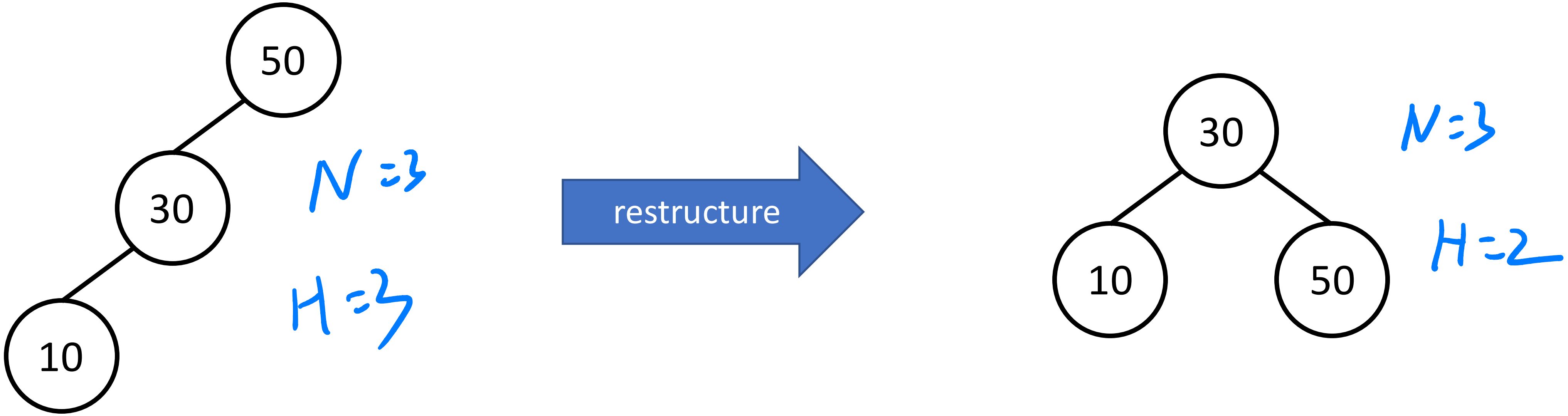
$\log_2(N+1) = H$

$\Rightarrow O(\log_2(N+\cancel{1}))$

$O(\log N)$

Binary Search Tree Rotations

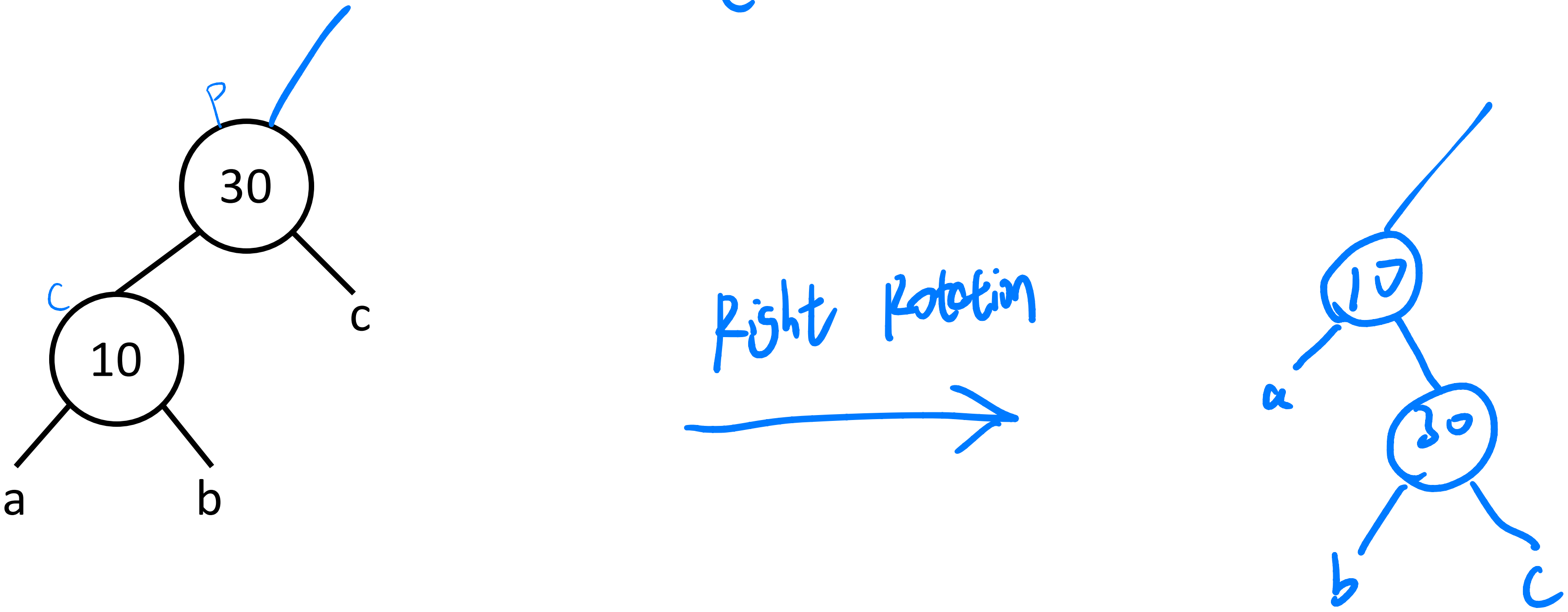
Goal: Manipulate Tree Structure



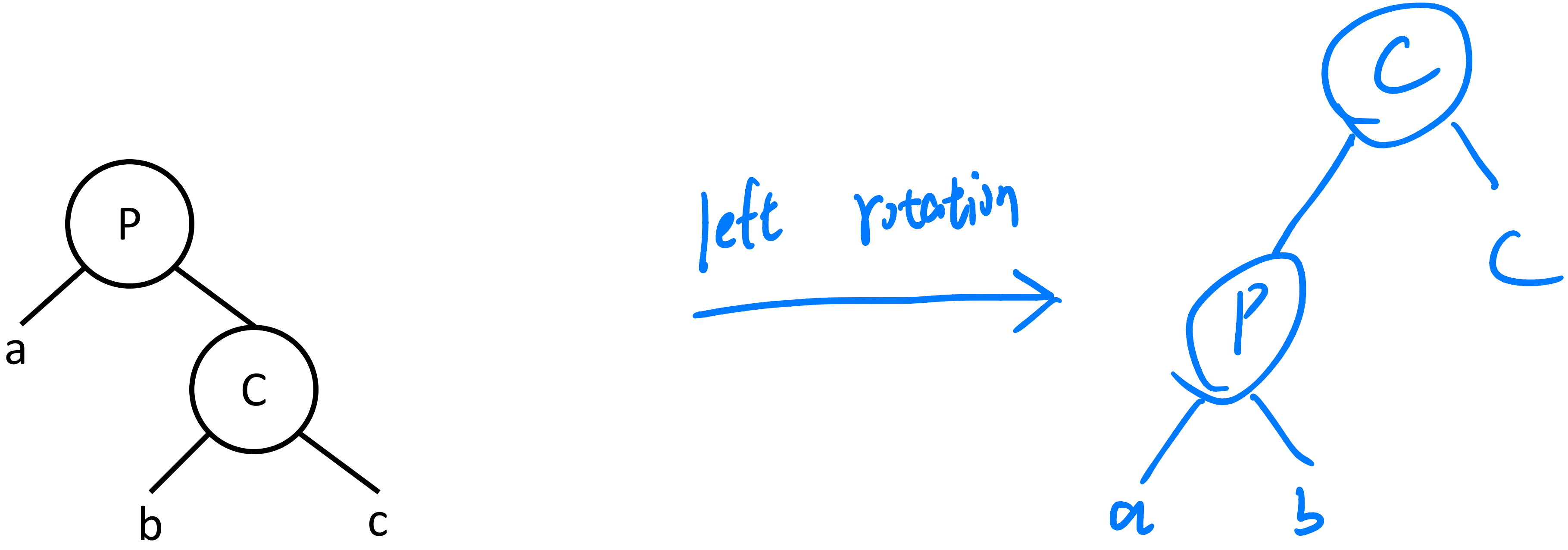
Right Rotation

P = parent

C = child

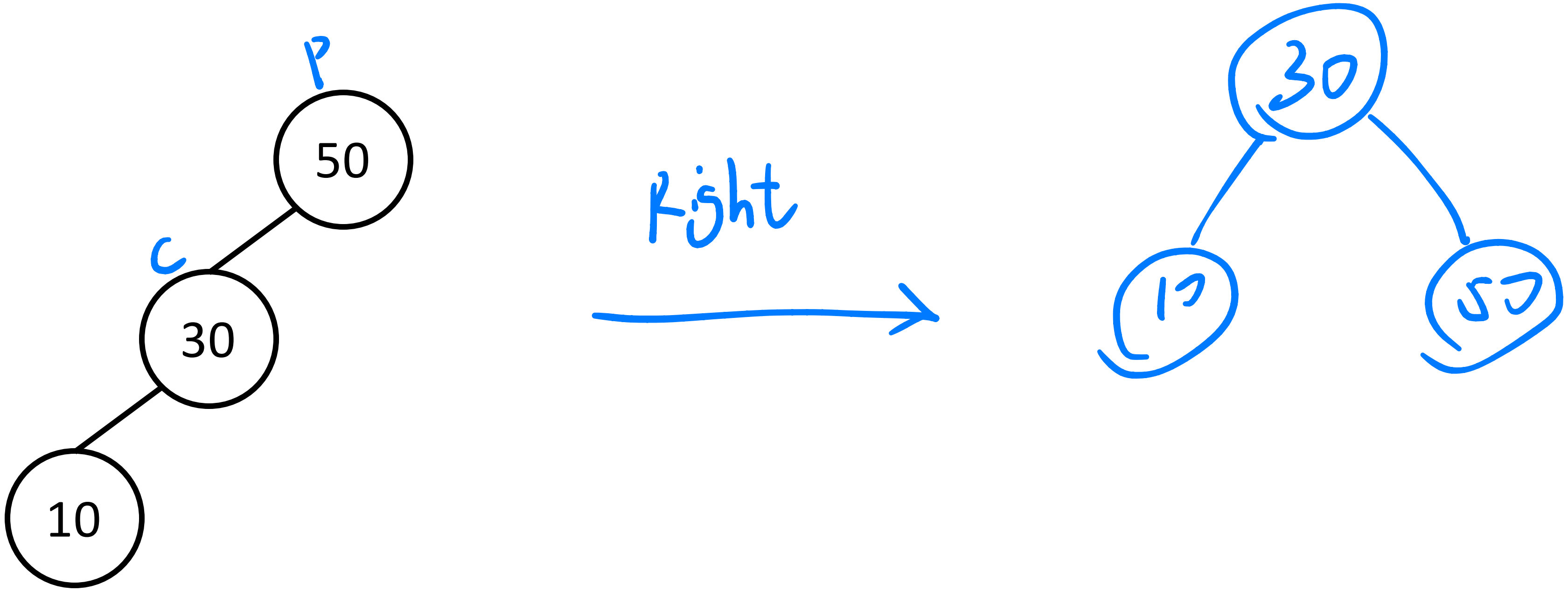


Left Rotation



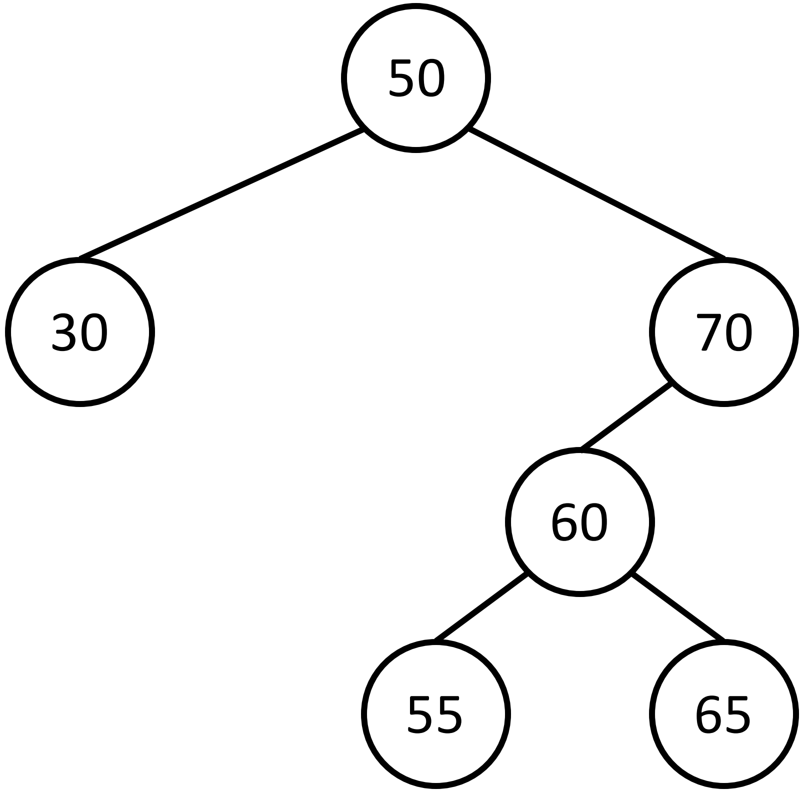
Rotation Practice

Rotate 50, 30

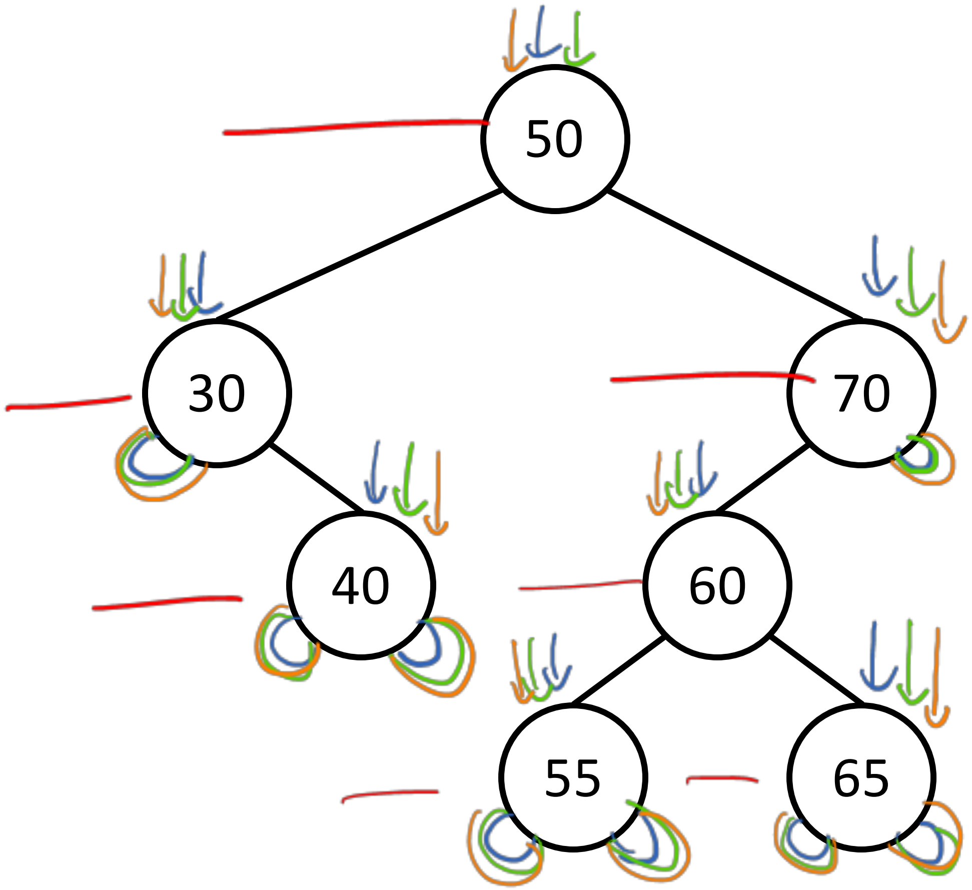


Rotation Practice

Rotate 70, 60



Tree Traversals



in-order: 30, 40, 50, 55, 60, 65, 70

pre-order: 50, 30, 40, 70, 60, 55, 65

post-order: 40, 30, 55, 65, 60, 70, 50

level-order: 50, 30, 70, 40, 60, 55, 65